Basic Observables for Processes*

Michele Boreale[1] Rocco De Nicola[2] Rosario Pugliese[2]

[1]Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza"
[2]Dipartimento di Sistemi e Informatica, Università di Firenze

Abstract. We propose a general approach to define behavioural preorders over process terms by considering the pre–congruences induced by three basic observables. These observables provide information about the initial communication capabilities of processes and about their possibility of engaging in an infinite internal chattering. We show that some of the observables–based pre–congruences do correspond to behavioral preorders long studied in the literature. The coincidence proofs shed light on the differences between the *must* preorder of De Nicola and Hennessy and the *fair/should* preorder of Cleaveland and Natarajan and of Brinksma, Rensink and Vogler, and on the rôle played in their definition by tests for internal chattering.

1 Introduction

In the classical theory of functional programming, the point of view is assumed that executing a program corresponds to evaluating it. If we write $M \downarrow v$ to indicate that program M evaluates to value v, the problem of the equivalence of two programs, hence of their semantics, can be stated as follows:

> Two programs M and N are *observationally equivalent* if for every program context C such that both $C[M]$ and $C[N]$ are programs, and for every value v, we have: $C[M] \downarrow v$ if and only if $C[N] \downarrow v$.

An alternative approach, used e.g. for the lazy lambda calculus [1], is that of defining a *simulation* (whose kernel is an equivalence) based on the reduction to normal forms. In general, given a language equipped with a reduction relation, the paradigm for defining equivalence over terms of the language, can be traced back to Morris [16] and can be phrased as follows:

1. Define a set of observables (values, normal forms, ...) to which a program can evaluate by means of successive reductions.
2. Consider the largest (pre–)congruence over the (set of operators of the) language induced by the chosen set of observables.

This paradigm has been the basis for assessing many semantics of sequential languages and is at the heart of the full abstraction problem, see e.g. [18].

Here, we aim at taking advantage of this paradigm also to assess models of concurrent systems and their equivalences. In this case, the choice of the basic observables is less obvious. On one hand, it is well–known that input/output

* Work partially supported by EEC: HCM project EXPRESS, by CNR project "Specifica ad alto livello e verifica formale di sistemi digitali" and by Istituto di Elaborazione dell'Informazione CNR, Pisa.

relations are not sufficient for describing the semantics of these classes of systems, and thus it would be limitative to use values as observables. On the other hand, studying the evolution to normal forms under all possible contexts is not as inspective as in the case of lambda calculus. Indeed, the interaction between a λ–term and the environment is circumscribed, while that between a process and its environment is less clear.

If we consider the λ–term MN, we know the extent of the influence of N over M, and, in any computation, we know exactly *when* an interaction between M and N occurs, namely when M reduces to a λ–abstraction. Thus by observing M in all possible contexts we can fully understand its behaviour. When considering concurrent systems, the internal evolution of each parallel component is freely intermingled with external communications. Then understanding the semantics of a component via its contextual behaviour turns out to be much less obvious.

Here, we shall consider a simple process description language, TCCS (Tau–less CCS [7]), and will study the impact of three basic observables for concurrent systems on this language. However, our results are easily extensible to general SOS language formats, like GSOS [2].

We shall be interested in testing for the *initial guaranteed* communication capabilities of a system. Indeed, when one is willing to infer the interactive behaviour of a system from its "isolated" behaviour, to know about the system's *possibility* of accepting communications along specific channels is not sufficient: due to the inherent nondeterminism of concurrent computations, it is necessary to know whether the acceptance of the communications is *guaranteed*. This is essential to establish liveness properties, like the absence of deadlock.

Moreover, we shall be interested in the risk a system has of getting involved in an infinite sequence of internal communications (to *diverge*), because this could lead to ignoring all subsequent external stimuli. Finally, with respect to this, it might also be important to know the external communications that can lead to divergent states.

These considerations guide us to introducing three basic observables:

1. $P\,!\,\ell$ (P *guarantees* ℓ) asserts that, by internal actions, P can only reach states from which action ℓ can be eventually performed;
2. $P\downarrow$ (P *converges*) asserts that P cannot get involved in an infinite sequence of internal actions;
3. $P\downarrow\ell$ (P *converges along* ℓ) asserts that P converges and does so also after performing ℓ.

For finite process graphs these observables are obviously decidable; in general, they are not, but this is somehow expected since the basic language (TCCS) is Turing powerful.

We shall analyze the impact of the above predicates on the semantics of TCCS. The predicates naturally induce five contextual preorders. These preorders are listed in Table 1; there we represent a contextual preorder using the notation ${}_{s_1}\preceq^{\mathrm{c}}_{s_2}$, where s_1 (if present) refers to the used convergence predicate, and s_2 (if present) refers to the guarantees one. The universal relation is denoted by $\mathcal{U}$.

conv./comm.	no req.	$\downarrow$	$\downarrow \ell$
no req.	$\mathcal{U}$	${}_{\downarrow}\preceq^{\mathrm{c}}$	${}_{\downarrow_{\mathcal{L}}}\preceq^{\mathrm{c}}$
$!\ell$	$\preceq^{\mathrm{c}}_{\mathcal{L}}$	${}_{\downarrow}\preceq^{\mathrm{c}}_{\mathcal{L}}$	${}_{\downarrow_{\mathcal{L}}}\preceq^{\mathrm{c}}_{\mathcal{L}}$

Table 1. Contextual Preorders

conv./comm.	no req.	$\downarrow$	$\downarrow \ell$
no req.	$\mathcal{U}$	$\sqsubseteq_{\widetilde{m}}$	$\sqsubseteq_{\widetilde{m}}$
$!\ell$	$\sqsubseteq_{\widetilde{FT}}$	$\sqsubseteq_{\widetilde{M}}$	$\sqsubseteq_{\widetilde{S}}$

Table 2. Main results

Our main results are five full abstraction theorems that make it manifest that our contextual preorders do coincide with well–known and/or intuitive behavioural preorders over processes studied in the literature. Table 2 provides a summary of the claimed results.

More specifically, we will show that:

- $\preceq^{\mathrm{c}}_{\mathcal{L}}$, the contextual preorder induced by $!\ell$, coincides with $\sqsubseteq_{\widetilde{FT}}$, the maximal pre–congruence included in the *fair/should* preorder of [17] and [3]. This pre–congruence can be characterized (see [4]) as the conjunction of the classical trace preorder (called *may* preorder in [6]) with the fair/should preorder;
- ${}_{\downarrow}\preceq^{\mathrm{c}}$ and ${}_{\downarrow_{\mathcal{L}}}\preceq^{\mathrm{c}}$, the contextual preorders induced by $\downarrow$ and $\downarrow \ell$, both coincide with $\sqsubseteq_{\widetilde{m}}$, the (reverse) inclusion of the *convergent traces* preorder, a simple variant of the trace preorder.

Together with the impact of the three observables used in isolation we also study the result of their conjunctions and show that:

- ${}_{\downarrow}\preceq^{\mathrm{c}}_{\mathcal{L}}$, the contextual preorder induced by $\downarrow$ and $!\ell$, coincides with $\sqsubseteq_{\widetilde{M}}$, the original *must* preorder of [6, 10];
- ${}_{\downarrow_{\mathcal{L}}}\preceq^{\mathrm{c}}_{\mathcal{L}}$, the contextual preorder induced by $\downarrow \ell$ and $!\ell$, gives rise to a *new* preorder, the *safe–must* preorder $\sqsubseteq_{\widetilde{S}}$, which is supported by a very intuitive testing scenario.

The safe–must preorder has a direct characterization in terms of computations from pairs of observers and processes: a computation is successful if a success state is reached *before* a catastrophic one (this explains the adjective 'safe'). This notion certainly deserves further investigation.

In the rest of the paper, we recall syntax and operational semantics (Sec. 2) and introduce an observational semantics (Sec. 3) for TCCS, then we present our full abstraction results (Sec. 4), compare the semantic preorders (Sec. 5) and briefly discuss related work. Due to space limitations, most proofs have been omitted; they can be found at `http://dsi2.dsi.unifi.it/~denicola`.

2 Tau–less CCS: TCCS

In this section, we briefly present the syntax and the operational semantics of TCCS, (τ–less CCS [7, 10]). We have preferred to use TCCS rather than CCS because it allows us to avoid the "congruence problems" that arise when the CCS choice operator (+) is used and silent actions are abstracted away. It is worth

mentioning that the very same results can be obtained by using CCS and its must pre–congruence obtained from the must preorder by imposing that whenever the "better" process can perform a silent move also the other can do it.

We assume an infinite set of *names* $\mathcal{N}$, ranged over by $a, b, \ldots$, and let $\overline{\mathcal{N}} = \{\overline{a} \mid a \in \mathcal{N}\}$, ranged over by $\overline{a}, \overline{b}, \ldots$, be the set of *co–names*. $\mathcal{N}$ and $\overline{\mathcal{N}}$ are disjoint and are in bijection via the *complementation* function $(\overline{\cdot})$; we define: $\overline{(\overline{a})} = a$. We let $\mathcal{L} = \mathcal{N} \cup \overline{\mathcal{N}}$, ranged over by $\ell, \ell', \ldots$, be the set of *labels*; we shall use B to range over subsets of $\mathcal{L}$ and we define $\overline{B} = \{\overline{\ell} \mid \ell \in B\}$. We also assume a countable set $\mathcal{X}$ of *process variables*, ranged over by $X, Y, \ldots$.

Definition 1. The set of *TCCS terms* is generated by the grammar:

$$E \;::=\; \mathbf{0} \;\mid\; \Omega \;\mid\; \ell.E \;\mid\; E[]F \;\mid\; E \oplus F \;\mid\; E \mid F \;\mid\; E\backslash L \;\mid\; E\{f\} \;\mid\; X \;\mid\; recX.E$$

where $f : \mathcal{L} \to \mathcal{L}$, called *relabelling function* is such that $\{\ell \mid f(\ell) \neq \ell\}$ is finite, $f(a) \in \mathcal{N}$ and $f(\overline{\ell}) = \overline{f(\ell)}$. We let $\mathcal{P}$, ranged over by P, Q, etc., denote the set of *closed* terms or *processes* (i.e. those terms where every occurrence of any agent variable X lies within the scope of some $recX.\_$ operator).

In the following, we often shall write ℓ instead of $\ell.\mathbf{0}$. We write $\_\{\ell'_1/\ell_1, \ldots, \ell'_n/\ell_n\}$ for the relabelling operator $\_\{f\}$ where $f(\ell) = \ell'_i$ if $\ell = \ell_i$, $i \in \{1, \ldots, n\}$, and $f(\ell) = \ell$ otherwise. As usual, we write $E[E_1/X_1, \ldots, E_n/X_n]$ for the term obtained by simultaneously substituting each occurrence of X_i in E with E_i (with renaming of bound process variables possibly involved).

The structural operational semantics of a TCCS term is defined via the two transition relations $\longrightarrow$ and $\succ\!\!\rightarrow$ induced by the inference rules in Table 3 and in Table 4, respectively. The symmetrical versions of rules `AR4` and `AR5` in Table 3 and of rules `IR5`, `IR6` and `IR7` in Table 4 have been omitted.

`AR1` $\ell.P \xrightarrow{\ell} P$	
`AR2` $\dfrac{P \xrightarrow{\ell} P'}{P\{f\} \xrightarrow{f(\ell)} P'\{f\}}$	`AR3` $\dfrac{P \xrightarrow{\ell} P'}{P\backslash L \xrightarrow{\ell} P'\backslash L}$ if $\ell \notin L \cup \overline{L}$
`AR4` $\dfrac{P \xrightarrow{\ell} P'}{P[]Q \xrightarrow{\ell} P'}$	`AR5` $\dfrac{P \xrightarrow{\ell} P'}{P \mid Q \xrightarrow{\ell} P' \mid Q}$

Table 3. SOS rules for TCCS: Action Relation

`IR1` $\Omega \succ\!\!\rightarrow \Omega$	`IR2` $recX.E \succ\!\!\rightarrow E[recX.E/X]$
`IR3` $\dfrac{P \succ\!\!\rightarrow P'}{P\{f\} \succ\!\!\rightarrow P'\{f\}}$	`IR4` $\dfrac{P \succ\!\!\rightarrow P'}{P\backslash L \succ\!\!\rightarrow P'\backslash L}$
`IR5` $P \oplus Q \succ\!\!\rightarrow P$	`IR6` $\dfrac{P \succ\!\!\rightarrow P'}{P \,[]\, Q \succ\!\!\rightarrow P' \,[]\, Q}$
`IR7` $\dfrac{P \succ\!\!\rightarrow P'}{P \mid Q \succ\!\!\rightarrow P' \mid Q}$	`IR8` $\dfrac{P \xrightarrow{\ell} P', \quad Q \xrightarrow{\overline{\ell}} Q}{P \mid Q \succ\!\!\rightarrow P' \mid Q'}$

Table 4. SOS rules for TCCS: Internal Relation

As usual, we use $\Longrightarrow$ or $\stackrel{\epsilon}{\Longrightarrow}$ to denote the reflexive and transitive closure of $\succ\!\!\rightarrow$ and use $\stackrel{s}{\Longrightarrow}$, with $s \in \mathcal{L}^+$, for $\Longrightarrow \stackrel{\ell}{\longrightarrow} \stackrel{s'}{\Longrightarrow}$ when $s = \ell s'$. Moreover, we write $P \stackrel{s}{\Longrightarrow}$ for $\exists P' : P \stackrel{s}{\Longrightarrow} P'$ ($P \stackrel{\ell}{\longrightarrow}$ and $P \succ\!\!\rightarrow$ will be used similarly). We will call *sort of* P the set $sort(P) = \{\ell \in \mathcal{L} \mid \exists s \in \mathcal{L}^* : P \stackrel{s\ell}{\Longrightarrow}\}$, *successors of* P the set $S(P) = \{\ell \in \mathcal{L} \mid P \stackrel{\ell}{\Longrightarrow}\}$, and *language* generated by P the set $L(P) = \{s \in \mathcal{L}^* \mid P \stackrel{s}{\Longrightarrow}\}$. Note that since we only consider finite relabelling operators, every TCCS process has a finite sort.

A *context* is a TCCS term C with one free occurrence of a process variable, usually denoted by $\_$. If C is a context, we write $C[P]$ instead of $C[P/\_]$. The context *closure* $\mathcal{R}^c$ of a given binary relation $\mathcal{R}$ over processes, is defined as: $P\,\mathcal{R}^c\,Q$ iff for each context C, $C[P]\,\mathcal{R}\,C[Q]$. $\mathcal{R}^c$ enjoys two important properties: (a) $(\mathcal{R}^c)^c = \mathcal{R}^c$, and (b) $\mathcal{R} \subseteq \mathcal{R}'$ implies $\mathcal{R}^c \subseteq \mathcal{R}'^c$. In the following, we will write $\not\mathcal{R}$ for the complement of $\mathcal{R}$.

3 Observational Semantics

In this section, we introduce different observational semantics for TCCS; we follow two approaches. The first approach takes advantage of basic observables, the second one of the classical *testing* scenario of [6, 10] and variants of it.

3.1 Basic Observables and Observation Preorders

Definition 2. Let P be a process and $\ell \in \mathcal{L}$. We define three basic *observation predicates* over processes as follows:

- $P\,!\,\ell$ (P *guarantees* ℓ) iff $\forall P' : P \Longrightarrow P'$ implies $P' \stackrel{\ell}{\Longrightarrow}$;
- $P \downarrow$ (P *converges*) iff there is no infinite sequence of internal transitions $P \succ\!\!\rightarrow P_1 \succ\!\!\rightarrow \cdots$ starting from P;
- $P \downarrow \ell$ (P *converges along* ℓ) iff $P \downarrow$ and $\forall P' : P \stackrel{\ell}{\Longrightarrow} P'$ implies $P' \downarrow$.

The above predicates can be combined in five sensible ways and used to define the corresponding basic *observation preorders* over processes, as stated in the following definition.

Definition 3. Let P and Q be processes.

- $P \;{}_{\downarrow}\preceq Q$ iff $P \downarrow$ implies $Q \downarrow$;
- $P \;{}_{\downarrow\mathcal{L}}\preceq Q$ iff for each $\ell \in \mathcal{L}$: $P \downarrow \ell$ implies $Q \downarrow \ell$;
- $P \preceq_{\mathcal{L}} Q$ iff for each $\ell \in \mathcal{L}$: $P\,!\,\ell$ implies $Q\,!\,\ell$;
- $P \;{}_{\downarrow}\preceq_{\mathcal{L}} Q$ iff for each $\ell \in \mathcal{L}$: $P \downarrow$ and $P\,!\,\ell$ implies $Q \downarrow$ and $Q\,!\,\ell$;
- $P \;{}_{\downarrow\mathcal{L}}\preceq_{\mathcal{L}} Q$ iff for each $\ell \in \mathcal{L}$: $P \downarrow \ell$ and $P\,!\,\ell$ implies $Q \downarrow \ell$ and $Q\,!\,\ell$.

Of course, the basic observation preorders are very coarse. More refined relations can be obtained by closing the above preorders under all TCCS contexts. For each basic observation preorder, say $\preceq$, the *contextual preorder* generated by $\preceq$ is defined as its closure $\preceq^c$.

3.2 Testing Preorders and Alternative Characterizations

Like in the original theory of testing [6, 10], we have that:

- *observers*, ranged over by $O, O', \ldots$, are processes capable of performing an additional distinct "success" action $w \notin \mathcal{L}$;
- *computations* from $P \mid O$ are sequences of internal transitions $P \mid O \succ\!\!\rightarrow P_1 \mid O_1 \succ\!\!\rightarrow \cdots$, which are either infinite or such that $P_k \mid O_k \not\succ\!\!\rightarrow$, $k \geq 0$.

Definition 4. Let P be a process and O be an observer.

1. $P \underline{must}_M O$ if for each computation from $P \mid O$, say $P \mid O \succ\!\!\rightarrow P_1 \mid O_1 \succ\!\!\rightarrow \cdots$, there is some $i \geq 0$ s.t. $O_i \stackrel{w}{\longrightarrow}$.
2. $P \underline{must}_S O$ if for each computation from $P \mid O$, say $P \mid O \succ\!\!\rightarrow P_1 \mid O_1 \succ\!\!\rightarrow \cdots$, there is some $i \geq 0$ s.t. $O_i \stackrel{w}{\longrightarrow}$ and $P_i \downarrow$.
3. $P \underline{must}_F O$ if for each computation from $P \mid O$, say $P \mid O \succ\!\!\rightarrow P_1 \mid O_1 \succ\!\!\rightarrow \cdots$, it holds that $P_i \mid O_i \stackrel{w}{\Longrightarrow}$ for each $i \geq 0$.

The first definition of successful computation given above is exactly that of [6]. The second one, considers successful only those computations in which a success state is reached *before* the observed process diverges. The third definition, which is essentially taken from [3], totally ignores the issue of divergence. These three notions allow us to define three preorders: the first one ($\sqsubseteq_{\approx M}$) is the original *must* preorder of [6, 10], the second one ($\sqsubseteq_{\approx S}$) is the new *safe–must* preorder and the third one ($\sqsubseteq_{\approx F}$) is the (reverse of the) *fair/should* preorder of [17] and [3].

Definition 5. Let $i \in \{M, S, F\}$. For all processes P and Q, $P \sqsubseteq_{\approx i} Q$ iff for every observer O: $P \underline{must}_i O$ implies $Q \underline{must}_i O$.

We introduce below alternative characterizations of the preorders *must* and *safe–must*. They support simpler methods for proving (or disproving) that two processes are behaviourally related. We need some additional notation.

Definition 6. Let $s \in \mathcal{L}^*$, $B \subseteq_{\text{fin}} \mathcal{L}$ and S be a set of processes.

- The *convergence* predicate, $\downarrow s$, is defined inductively as follows: $P \downarrow \epsilon$ if $P \downarrow$; $P \downarrow \ell s'$ if $P \downarrow \epsilon$ and $\forall P' : P \stackrel{\ell}{\Longrightarrow} P'$ implies $P' \downarrow s'$.
  We write $P \uparrow s$ if $P \downarrow s$ does not hold.
- $(P \underline{after} s)$ denotes the set of processes $\{P' : P \stackrel{s}{\Longrightarrow} P'\}$.
- We write $P \downarrow B$ if $\forall \ell \in B : P \downarrow \ell$ and $S \downarrow B$ if $\forall P \in S : P \downarrow B$.
- $P \,!\, B$ stands for $\forall P' : P \Longrightarrow P'$ implies $\exists \ell \in B : P' \stackrel{\ell}{\Longrightarrow}$.
- $S \downarrow\!! B$ stands for $\forall P \in S : P \downarrow B$ and $P \,!\, B$.

Definition 7. For all processes P and Q, we write

- $P \ll_M Q$ if $\forall s \in \mathcal{L}^*$ such that $P \downarrow s$, it holds that:
  (a) $Q \downarrow s$, and (b) for every $B \subseteq_{\text{fin}} \mathcal{L}$: $(P \underline{after} s) \,!\, B$ implies $(Q \underline{after} s) \,!\, B$.
- $P \ll_S Q$ is the same as above but predicate $!$ is replaced by $\downarrow\!!$.

Theorem 8. For all processes P and Q, (1) $P \sqsubseteq_{\approx_M} Q$ iff $P \ll_M Q$ and (2) $P \sqsubseteq_{\approx_S} Q$ iff $P \ll_S Q$.

By taking advantage of the above alternative characterizations it is easy to prove that the must and the safe–must preorders are pre–congruences.

Theorem 9. For all processes P and Q and $i \in \{M, S\}$, $P \sqsubseteq_{\approx_i} Q$ iff $P \sqsubseteq^{\mathrm{c}}_{\approx_i} Q$.

Note that the congruence result does not hold for the fair/should preorder $\sqsubseteq_{\approx_F}$, it is not preserved by the recursion operator. This can be easily seen by considering the following counter–example. Consider the processes $P = a.b[]a.c$ and $Q = a.b$ and the context $C = recX.(\_ | \overline{a}.\overline{b}.X)\backslash\{a, b\}$. It obviously holds that $P \sqsubseteq_{\approx_F} Q$, but $C[P] \not\sqsubseteq_{\approx_F} C[Q]$ (just take $O = \overline{c}.w$); hence $P \not\sqsubseteq^{\mathrm{c}}_{\approx_F} Q$.

An alternative characterization of the closure of the fair/should preorder is given in [4], for a language slightly different from ours.

Definition 10. For all processes P and Q, we write

$$P \sqsubseteq_{\approx_{FT}} Q \text{ if } (P \sqsubseteq_{\approx_F} Q \text{ and } L(P) \subseteq L(Q)).$$

Theorem 11. For all processes P and Q, $P \sqsubseteq_{\approx_{FT}} Q$ iff $P \sqsubseteq^{\mathrm{c}}_{\approx_F} Q$.

4 Full Abstraction Results

From now on, we adopt the following convention: an action declared *fresh* in a statement is supposed to be different from any other name and co–name mentioned in the statement.

4.1 Convergence predicate and convergent traces

In this section, we deal with the first two contextual preorders, $\preceq^{\mathrm{c}}_{\downarrow}$ and $\preceq^{\mathrm{c}}_{\downarrow_{\mathcal{L}}}$, and prove that they have the same distinguishing power and coincide with the reverse inclusion of the convergent traces preorder.

Definition 12. For all processes P and Q, we write $P \sqsubseteq_{\approx_m} Q$ if $\forall s \in \mathcal{L}^*$ such that $P \downarrow s$, it holds that:
a) $Q \downarrow s$, and
b) $s \in L(Q)$ implies $s \in L(P)$.

Theorem 13. For all processes P and Q, $P \sqsubseteq_{\approx_m} Q$ iff $P \sqsubseteq^{\mathrm{c}}_{\approx_m} Q$.

The following special contexts can be used to prove the next theorems. If $s \in \mathcal{L}^*$, say $s = \ell_1 \cdots \ell_n$ $(n \geq 0)$, we define
- $C^s_1 = \_ | \overline{\ell}_1.\cdots.\overline{\ell}_n.\mathbf{0}$ and
- $C^s_2 = \_ | \overline{\ell}_1.\cdots.\overline{\ell}_n.\Omega$.

Theorem 14. For all processes P and Q, $P \sqsubseteq_{\approx_m} Q$ iff $P \preceq^{\mathrm{c}}_{\downarrow} Q$.

Theorem 15. For all processes P and Q, $P \preceq^{\mathrm{c}}_{\downarrow_{\mathcal{L}}} Q$ iff $P \preceq^{\mathrm{c}}_{\downarrow} Q$.

4.2 Guarantees and fair testing

Lemma 16. Let P be a process, O be an observer and let $\ell \in \mathcal{L}$ be a fresh action; (1) $P \underline{must}_F O$ iff $P \mid O\{\ell/w\} ! \ell$, and (2) $P ! \ell$ iff $P \underline{must}_F \overline{\ell}.w$.

Theorem 17. For all processes P and Q, $P \sqsubseteq^{\mathrm{c}}_{\approx_F} Q$ iff $P \preceq^{\mathrm{c}}_{\mathcal{L}} Q$.

PROOF: ($\Longleftarrow$) We prove that $\preceq^{\mathrm{c}}_{\mathcal{L}}$ is contained in $\sqsubseteq^{\mathrm{c}}_{\approx_F}$, the claimed result follows by closing under contexts. Suppose that $P \preceq^{\mathrm{c}}_{\mathcal{L}} Q$ and that $P \underline{must}_F O$; let ℓ be a fresh action. We have:

$$\begin{array}{rl} P \underline{must}_F O & \text{implies (Lemma 16(1))} \\ P \mid O\{\ell/w\} ! \ell & \text{implies (hypothesis } P \preceq^{\mathrm{c}}_{\mathcal{L}} Q, \text{ with } C = \_ \mid O\{\ell/w\}) \\ Q \mid O\{\ell/w\} ! \ell & \text{implies (Lemma 16(1))} \\ Q \underline{must}_F O & \end{array}$$

($\Longrightarrow$) The proof is similar but relies on Lemma 16(2). □

4.3 Guarantees and convergence, and must testing

The next definition introduces two special contexts to be used in the proof of Theorem 20.

Definition 18. Let $s \in \mathcal{L}^*$, say $s = \ell_1 \cdots \ell_n$ ($n \geq 0$), and $B \subseteq_{\mathrm{fin}} \mathcal{L}$. Let f^B denote a function which maps each $\ell \in B$ to a single fresh c. Fix a bijective correspondence among $\ell_1, \ldots, \ell_n$ and n fresh actions $\alpha_1, \ldots, \alpha_n$. We define

- $C_3^s = \_ \mid Q_3^s$ where $Q_3^\epsilon = c$ and $Q_3^{\ell s'} = \overline{\ell}.Q_3^{s'} [] c$, and
- $C_4^{s,B} = (\_ \mid R^s)\{f^B\} \mid Q_4^s$ where $R^s = \overline{\ell}_1.\alpha_1. \cdots \overline{\ell}_n.\alpha_n$, $Q_4^\epsilon = \mathbf{0}$ and $Q_4^{\ell_1 s'} = \overline{\alpha}_1.Q_4^{s'} [] c$.

Lemma 19. Let $s \in \mathcal{L}^*$, $B \subseteq_{\mathrm{fin}} \mathcal{L}$ and c be a fresh action.
a) $P \downarrow s$ iff $C_3^s[P] \downarrow$ iff $C_3^s[P] \downarrow c$.
b) $(P \underline{after} s) ! B$ iff $C_4^{s,B}[P] ! c$.

Theorem 20. For all processes P and Q, $P \sqsubseteq_{\approx_M} Q$ iff $P \,{}_{\downarrow}\preceq^{\mathrm{c}}_{\mathcal{L}} Q$.

PROOF: ($\Longrightarrow$) From the definition, it is easily seen that $\ll_M$ is contained in ${}_{\downarrow}\preceq_{\mathcal{L}}$ (indeed $P ! c$ iff $(P \underline{after} \epsilon) ! \{c\}$). From this fact, by closing under contexts and applying Theorem 8, the thesis follows.

($\Longleftarrow$) Here, we show that ${}_{\downarrow}\preceq^{\mathrm{c}}_{\mathcal{L}}$ is contained in $\ll_M$. From this fact and Theorem 8, the thesis follows. Assume that $P \,{}_{\downarrow}\preceq^{\mathrm{c}}_{\mathcal{L}} Q$ and that $P \downarrow s$, for some $s \in \mathcal{L}^*$. We have to show that: (a) $Q \downarrow s$ and (b) $(P \underline{after} s) ! B$ implies $(Q \underline{after} s) ! B$, for any $B \subseteq_{\mathrm{fin}} \mathcal{L}$. As to part (a), from $P \downarrow s$ and Lemma 19(a), it follows that $C_3^s[P] \downarrow$. Obviously, for every process R, $C_3^s[R] ! c$. From $C_3^s[P] \downarrow$, $C_3^s[P] ! c$ and $P \,{}_{\downarrow}\preceq^{\mathrm{c}}_{\mathcal{L}} Q$ it follows that $C_3^s[Q] \downarrow$. By applying again Lemma 19(a), but in the reverse direction, we obtain $Q \downarrow s$. As to part (b), suppose that $(P \underline{after} s) ! B$. From this, applying Lemma 19(b), it follows that $C_4^{s,B}[P] ! c$. Moreover, it is easy to see that for every process R, $R \downarrow s$ implies $C_4^{s,B}[R] \downarrow$. From $C_4^{s,B}[P] \downarrow$, $C_4^{s,B}[P] ! c$ and $P \,{}_{\downarrow}\preceq^{\mathrm{c}}_{\mathcal{L}} Q$, it follows that $C_4^{s,B}[Q] ! c$. By applying again Lemma 19(b), but in the reverse direction, we obtain $(Q \underline{after} s) ! B$. □

4.4 Guarantees and convergence, and safe–must

To prove full abstraction for safe–must, we will use another special context. Again, we assume that $c \in \mathcal{L}$ is always fresh. If $s \in \mathcal{L}^*$, say $s = \ell_1 \cdots \ell_n$ $(n \geq 0)$, and $B \subseteq_{\text{fin}} \mathcal{L}$, we define the context

– $C_5^{s,B} = \_ \mid Q_5^{s,B}$ where $Q_5^{\epsilon,B} = \sum_{\ell \in B} \overline{\ell}.c$ and $Q_5^{\ell s',B} = \overline{\ell}.Q_5^{s',B} [] c$.

The proof of the following theorem is similar to that of Theorem 20, but relies on the context $C_5^{s,B}$ instead of $C_4^{s,B}$.

Theorem 21. For all processes P and Q, $P \sqsubseteq_S Q$ iff $P \;{}_{\downarrow\mathcal{L}}\preceq^{\mathrm{c}}_{\mathcal{L}}\; Q$.

It is worthwhile to point out why the context $C_5^{s,B}$ cannot be used in place of the context $C_4^{s,B}$ to prove full abstraction for the must preorder (Theorem 20). Indeed, $P \downarrow s$ does not imply that $C_5^{s,B}[P] \downarrow$ (for instance $a.b.\Omega \downarrow a$ but $C_5^{a,\{b\}}[a.b.\Omega] \uparrow$). This would invalidate the proof of the "if" part of Theorem 20.

5 Comparing the preorders

Theorem 22. For all processes P and Q, $P \sqsubseteq_M Q$ implies $P \sqsubseteq_S Q$, but not vice–versa.

Proof: Paralleling the proof of Theorem 20, part $\Longleftarrow$, it is easy to show that $_{\downarrow}\preceq^{\mathrm{c}}_{\mathcal{L}}$ is contained in $\ll_S$, from which the result will follow by applying Theorems 20 and 8. To show that the vice–versa does not hold, consider $P \stackrel{\text{def}}{=} a.b.\Omega$ and $Q \stackrel{\text{def}}{=} a$. It is easy to see that $P \sqsubseteq_S Q$, but $P \;{}_{\downarrow}\not\preceq^{\mathrm{c}}_{\mathcal{L}}\; Q$ (just consider $\_ \mid \overline{a}$). □

Theorem 23.

1. $\sqsubseteq_M \;=\; {}_{\downarrow}\preceq^{\mathrm{c}}_{\mathcal{L}} \;\subset\; {}_{\downarrow\mathcal{L}}\preceq^{\mathrm{c}}_{\mathcal{L}} \;=\; \sqsubseteq_S \;\subset\; {}_{\downarrow\mathcal{L}}\preceq^{\mathrm{c}} \;=\; {}_{\downarrow}\preceq^{\mathrm{c}}$.
2. $\preceq^{\mathrm{c}}_{\mathcal{L}} \;=\; \sqsubseteq_{FT}$ and $\sqsubseteq_{FT}$ is not comparable with $\sqsubseteq_M$, $\sqsubseteq_S$ and ${}_{\downarrow}\preceq^{\mathrm{c}}$.

Proof:

1. The result follows from Theorems 14, 20, 21 and 22. By definition, it is easily seen that $_{\downarrow\mathcal{L}}\preceq^{\mathrm{c}}_{\mathcal{L}}$ is included in $_{\downarrow\mathcal{L}}\preceq^{\mathrm{c}}$. The inclusion is strict: $a \;{}_{\downarrow\mathcal{L}}\preceq^{\mathrm{c}}\; \mathbf{0}$ but $a \;{}_{\downarrow\mathcal{L}}\not\preceq_{\mathcal{L}}\; \mathbf{0}$.
2. The equality $\preceq^{\mathrm{c}}_{\mathcal{L}} \;=\; \sqsubseteq_{FT}$ derives from Theorems 17 and 11. To see that neither of $\sqsubseteq_M$, $\sqsubseteq_S$ and ${}_{\downarrow}\preceq^{\mathrm{c}}$ is included in $\sqsubseteq_F$ (hence in $\sqsubseteq_{FT}$), consider the processes $P \stackrel{\text{def}}{=} recX.(a.X [] a.b)$ and $Q \stackrel{\text{def}}{=} recX.a.X$. Clearly, $P \sqsubseteq_M Q$, hence $P \sqsubseteq_S Q$ and $P \;{}_{\downarrow\mathcal{L}}\preceq^{\mathrm{c}}\; Q$. However, $P \not\sqsubseteq_F Q$ (because $P \;\underline{must}_F\; O$ and $Q \;\underline{m\not{u}st}_F\; O$, when $O \stackrel{\text{def}}{=} recX.(\overline{a}.X [] \overline{b}.w)$). To see the converse, observe that $\mathbf{0} \sqsubseteq_{FT} \Omega$, but $\mathbf{0} \;{}_{\downarrow}\not\preceq\; \Omega$, hence $\mathbf{0} \not\sqsubseteq_S \Omega$ and $\mathbf{0} \not\sqsubseteq_M \Omega$. □

The mutual relationships among the pre–congruences are simpler if we move to *strongly convergent* processes. We say that a process P is strongly convergent if $P \downarrow s$ for every $s \in \mathcal{L}^*$.

Theorem 24. For strongly convergent processes, it holds that:

$$\preceq^{\mathrm{c}}_{\mathcal{L}} \;=\; \sqsubseteq_{FT} \;\subset\; \sqsubseteq_M \;=\; {}_{\downarrow}\preceq^{\mathrm{c}}_{\mathcal{L}} \;=\; {}_{\downarrow\mathcal{L}}\preceq^{\mathrm{c}}_{\mathcal{L}} \;=\; \sqsubseteq_S \;\subset\; {}_{\downarrow\mathcal{L}}\preceq^{\mathrm{c}} \;=\; {}_{\downarrow}\preceq^{\mathrm{c}}.$$

6 Conclusions

We have proposed three basic notions of process observables, that, when closed with respect to the contexts of a CCS–like language, induce five pre–congruences that have been proved to coincide with well–known and/or intuitive behavioural relations.

Notions of observables in the same spirit as ours have been proposed in [13], [21], [11], [15], [8] and [12].

In [13], it is shown that the pre–congruence induced by inclusion of maximal traces coincides, both for CCS and CSP, with the must pre–congruence of [6]; another characterization is given by only considering the inclusion of the maximal ϵ–trace, i.e. a sequence of invisible moves leading to a divergent state or to a deadlocked one. The strength of the basic observables (maximal traces are definitely more inspective than our guarantees predicate) prevents from capturing different notions such as fair testing, and hinders the rôle played by the convergence test, which is somehow included in that for maximality.

In [21], two Petri nets are called *d–equivalent* if they both can reach a deadlocked state or if they both cannot do so. Then it is proved that, by closing d–equivalence with respect to parallel composition, the variant of failure semantics [5] that ignores divergence is obtained.

In [11], a series of variants of the testing framework is proposed and results are listed showing that, by changing the expressive power of testers, a number of equivalences ranging from bisimulation to testing can be captured. One of the considered family of observers is that consisting just of agents of the form $\ell.w.\mathbf{0}$, that somehow resemble our $!\ell$ predicates. It is claimed that for strongly convergent processes the pre–congruence induced by this family of observers coincides with the must preorder and the reader is referred to [13] for the proof. However, we could not find the proof in Main's paper.

Milner and Sangiorgi [15] define an equivalence for processes based on elementary observables, namely the *possibility* for a process to synchronize along a specific channel. However, they permit to recursively test for the presence of this observable. The resulting notion of observability (called barbed bisimilarity), when closed under parallel composition, yields bisimulation–based equivalences that are significantly more discriminating than ours.

Ferreira [8] and Laneve [12] deal with languages significantly different from classical process algebras. In particular, Ferreira uses a predicate which resembles very much the conjunction of our $\downarrow$ and $!\,\ell$ (based on production of values rather than on communication capabilities) to define a testing preorder for Concurrent ML [20]; this seems to be strongly related to our safe–must preorder. He also conjectures that if one considers pure CCS (and observes communication capabilities instead of value productions) the obtained preorder coincides with the *must* pre–congruence of [6]; here we have proved this conjecture. Laneve discusses the impact of an observables-based testing scenario on the Join Calculus, a language with elaborate synchronization schemata [9].

Acknowledgments

We are grateful to L. Aceto, F. van Breugel, W. Ferreira, A. Rensink and W. Vogler for interesting discussions and suggestions and to F. Focardi for a first debugging of the ideas contained in the paper.

References

1. S. Abramsky. The lazy lambda calculus. *Research Topics in Functional Programming*, David Turner, ed., Addison–Wesley, 1990.
2. B. Bloom, S. Istrail, A.R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232-268, 1995.
3. E. Brinksma, A. Rensink, W. Vogler. Fair Testing. *Proceedings of CONCUR'95*, *LNCS* 962, pages 313-327, Springer, 1995.
4. E. Brinksma, A. Rensink, W. Vogler. Applications of Fair Testing. In R. Gotzhein and J. Bredereke, ed., Formal Description Techniques IX, Chapman & Hall, 1996.
5. S.D. Brookes, C.A.R. Hoare, A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560-599, 1984.
6. R. De Nicola, M.C.B. Hennessy. Testing Equivalence for Processes. *Theoretical Computers Science*, 34:83-133, 1984.
7. R. De Nicola, M.C.B. Hennessy. CCS without τ's. *Proceedings of TAPSOFT'87*, *LNCS* 249, pages 138-152, Springer, 1987.
8. W. Ferreira. *Semantic Theories for Concurrent ML*. Ph.D. Thesis, University of Sussex, 1996.
9. C. Fournet, G. Gonthier, J.-L. Lévy, L. Maranget, D. Rémy. A Calculus of Mobile Agents. *Proceedings of CONCUR'96*, *LNCS* 1119, 1996.
10. M.C.B. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.
11. M.C.B. Hennessy. Observing Processes. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, *LNCS* 354, Springer, 1989.
12. C. Laneve. May and Must Testing in the Join-Calculus. Technical Report UBLCS-96-4, Università di Bologna, Dept. of Computer Science, Bologna, 1996.
13. M.G. Main. Trace, Failure and Testing Equivalences for Communicating Processes. *Int. Journal of Parallel Programming*, 16(5):383-400, 1987.
14. R. Milner. *Communication and Concurrency*. Prentice Hall International, 1989.
15. R. Milner, D. Sangiorgi. Barbed Bisimulation. *Proceedings of ICALP'92*, *LNCS* 623, Springer, 1992.
16. J.-H. Morris. *Lambda Calculus Models of Programming Languages*. Ph.D. Thesis, MIT, 1968.
17. V. Natarajan, R. Cleaveland. Divergence and Fair Testing. *Proceedings of ICALP'95*, *LNCS* 944, pages 648-659, Springer, 1995.
18. C.-H.L. Ong. Correspondence between operational and denotational semantics: the full abstraction problem for PCF. *Handbook of Logic in Computer Science*, vol.4, S. Abramsky, D.M. Gabbay and T.S.E. Maibaum, ed., Oxford Science Publ., 1995.
19. G.D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, Aarhus University, Dept. of Computer Science, Aarhus, 1981.
20. J.H. Reppy. Concurrent ML: Design, application and semantics. *Proceedings of Functional Programming, Concurrency, Simulation and Automata Reasoning*, *LNCS* 693, pages 165-198, Springer, 1993.
21. W. Vogler. Failures Semantics and Deadlocking of Modular Petri Nets. *Acta Informatica*, 26:333-348, 1989.